AE 353 Project 3: Satellite and Star-Tracker

Anshuk Chigullapalli[1]
University of Illinois at Urbana-Champaign, Champaign, Illinois, 61820, USA

The goal of this project is to design, implement and test a controller that controls the four reaction wheels of a satellite to maintain its initial attitude, and an observer that collects data on the position of stars and estimates the state vector of the system. This report will discuss the design process, features and shortcomings of the controller and the observer. All the simulations were designed and run in a Jupyter notebook environment.

I. Nomenclature

A, B, C	=	Matrices that define the state space model
α	=	Angle of ascension of a star
δ	=	Angle of declination of a star
F	=	Equations of motion for the controller model
G	=	Equations of motion for the sensor model
G_{eq}	=	Equilibrium value of G
K	=	Controller gain matrix
L	=	Observer gain matrix
MSE	=	Mean squared error
ϕ	=	Roll angle of the satellite
ψ	=	Yaw angle of the satellite
Q_c, R_c	=	Cost matrices for controller LQR
Q_o, R_o	=	Cost matrices for observer LQR
u	=	The vector of inputs of the system
θ	=	Pitch angle of the satellite
τ_1 to τ_4	=	Torque inputs to the reaction wheels
$\omega_x, \omega_y, \omega_z$	=	Angular rates of the satellite
W_c	=	Controllability matrix
W_o	=	Observability matrix
x	=	State vector
y	=	Output vector

II. Introduction

Star trackers are high-end sensors used often in the navigation of satellites in orbit. Star trackers observe the positions of the different stars in their field of view, and using this information determine the position and orientation of the satellite they are harnessed to. These devices come in many different sizes, from massive scopes used on large satellites, such as the one shown in Figure 1, to ones that can fit on small CubeSats.

[1] Student, Department of Aerospace Engineering.

Figure 1: A high accuracy star tracker made by Ball Aerospace

In the system model studied in this project, the satellite has a star tracker scope that tracks the position of stars on a spherical surface that is supposed to imitate an approximate "night sky".

Star trackers themselves only provide navigation; they have no means of controlling the satellite themselves. In this project, the satellite has four reaction wheels. These reaction wheels provide full controllability (proved in the Jupyter notebook and highlighted in Section (IV.B) of the satellite's orientation and angular velocities.

Shown in is a snapshot of the satellite and the star arrangement used in this project. There are 5 stars in this scenario.

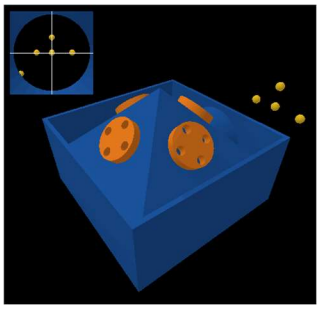

Figure 2: Satellite and star tracker

The system is also subject to many random disturbances, such as random initial conditions and random "space debris" that can hit the satellite and change its orientation. The sections of this report go over the design, implementation, testing and analysis of the results of this observer-controller control system.

III.Requirements and Validation

The efficacy of the designed observer and controller system cannot be tested with only a single iteration of the simulation. This is because the system is subjected to random behavior that can result in different results in every simulation run. Therefore, the requirements of the simulation are based on aggregate results from multiple simulations.

This set of requirements is by no means comprehensive and are not rigorous enough for a control system that would be implemented on a real satellite. However, due to the constraints on the simulation and the author's time, this set of requirements was considered adequate:

Requirement: With normally-distributed-random initial orientation and sensor noise setting of 0.1, zero initial angular velocities, and random disturbances, the spacecraft should be capable of orienting itself as close to an orientation of zero roll ($\phi_{des} = 0$), zero pitch ($\theta_{des} = 0$) and zero yaw ($\psi_{des} = 0$) as possible. The perfect result is rarely expected, and so the mean squared error (for a 30 second simulation) of the roll must be less than $0.5\ rad^2$, and the mean square error of the pitch and yaw must be less than $0.05\ rad^2$, for at least 75% of all the simulations run.

It is important to note that the simulations are run with zero initial angular velocities, rather than random initial angular velocities. The requirement is a test of both the controller and observer rather than just one of them, and it directly measures the variables that we want controlled. No controller would work 100% of the time because of the random initial conditions and space debris, and so the 75% accuracy is to provide a reasonable yet useful benchmark to meet.

These requirements have to be tested and validated. For this, we set up the following verification procedure:

- Pybullet will be used to simulate the system. The code will be run in a Jupyter notebook environment. The simulation will be run 500 times, to gather adequate data for analysis.
- In each simulation, the mean square error of the roll, pitch and yaw will be calculated and stored in an array. The mean square error of a quantity x is defined as shown in the equation below:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(x - x_{des})^2$$

- Since the desired values for roll, pitch and yaw are zero, the error is the same as the value of the orientation angle itself.
- A histogram of the mean squared errors of all the simulations will be created. This can be observed to see the general behavior and validity of the system.
- The 75th percentile will be found using numpy's quantile function. This gives us the worst result of the top 75% of the simulations, which implies that the system has a better result than the 75th percentile at least 75% of the time.

With the requirements and verification method set, the observer and controller can now be designed and tested.

IV. Model

A. Controller Equations of Motion and Linearization

The purpose of the system is to be able to control orientation. Therefore, the state vector x only includes the orientation angles: roll(ϕ), pitch (θ), and yaw (ψ). It also includes the angular velocities of the body ($\omega_x, \omega_y, \omega_z$) as states for better control. Our satellite has four reaction wheels, so we have four torque inputs in our input matrix.

With this, the state and input of the system are defined as follows:

$$x = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \qquad u = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} \tag{1}$$

Since the equilibrium values are all zero, they are not included in the state and input matrices above. The equations of motion for this system were provided with the design problem. These equations of motion are placed a matrix f such that $\dot{x} = f$.

We linearize these equations of motion about an equilibrium point. For this control system, the goal is to minimize the roll, pitch, yaw, and angular velocities. So, all the states have an equilibrium value of zero. At this equilibrium point, $f = 0$, therefore the point is valid. Using this equilibrium point and f, we linearize the system to get A and B.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -0.0524 & 0.0524 & 0 & 0 \\ 0 & 0 & -0.0524 & 0.0524 \\ -0.0373 & -0.0373 & -0.0373 & -0.0373 \end{bmatrix} \tag{4}$$

We now have a linearized system of the state space form shown below.

$$\dot{x} = Ax + Bu \tag{5}$$

B. Checking for Controllability

Before we design a controller, we need to check whether our system is controllable. To do so, we find the controllability matrix W. It is defined as shown below:

$$W_c = [B\ AB\ A^2B\ A^3B\ \dots A^{n-1}B] \tag{6}$$

n is equal to the number of states. In this case, $n = 6$. The controllability matrix is calculated in the python notebook. The rank of the controllability matrix is what determines whether the system is controllable or not. If the rank is equal to the number of states in the system, then the system is controllable. The rank was found using the numpy.linalg.matrix_rank() function. We see that W has a rank of 6, which means that our system, for our choice of states, is controllable.

C. Observer Equations of Motion and Linearization

We are given the equations of motion that define the sensor measurements as a function of the satellite's orientation and the position of the star in the sky. Since we have five stars, we have 10 measurements in our output matrix y. We can place all the equations of motion in a matrix G, such that G is the nonlinear version of y.

$$G = g(\phi, \theta, \psi, \alpha_1, \delta_1, \alpha_1, \delta_1, \alpha_2, \delta_2, \alpha_3, \delta_3, \alpha_4, \delta_4, \alpha_5, \delta_5) \tag{7}$$

The ascension and declination of the five stars chosen are given below:

$$(\alpha, \delta) = (0,0), (0.15, 0), (0, 0.15), (-0.2, 0), (0.3, -0.2) \tag{8}$$

These values are substituted in place of the α and δ variables shown in Equation (7).

Unlike f, G does not evaluate to zero at the equilibrium point. The value that it evaluates will be called G_{eq}. This value is important as it must be subtracted from the sensor measurements to get the true output.

$$y = \begin{bmatrix} y_1 \\ z_1 \\ \vdots \\ y_5 \\ z_5 \end{bmatrix} - G_{eq} \tag{9}$$

G itself can be linearized about the equilibrium value to find C. The calculated value of C is shown below:

$$C = \begin{bmatrix} 0 & 0 & -2.625 & 0 & 0 & 0 \\ 0 & 2.625 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2.6849 & 0 & 0 & 0 \\ -0.3967 & 2.625 & 0 & 0 & 0 & 0 \\ 0.3967 & 0 & -2.625 & 0 & 0 & 0 \\ 0 & 2.6849 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2.7328 & 0 & 0 & 0 \\ 0.53311 & 2.625 & 0 & 0 & 0 & 0 \\ -0.55699 & -0.1723 & -2.876 & 0 & 0 & 0 \\ -0.812 & 2.7432 & 0.1723 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

We know have the sensor model in the form of the state space equation:

$$y = Cx \tag{11}$$

D. Checking for Observability

Just like with controllability, we need to ensure that the system we've selected is observable. For this, we determine the observability matrix and check its rank. The observability matrix is defined as follow:

$$W_o = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{12}$$

n is the number of states. The observability matrix is calculated in the jupyter notebook and its rank is calculated using numpy's functions. The rank for the calculated W_o is 6, which is equal to the number of states, which means that the system is observable.

V. Design

We need to design an observer to find the estimated state of the system, and a controller to find the input to the system. In control system design, it is proven that the observer and controller can be designed separately and still be implemented together.[2]

The observer gain L and the controller gain K are defined by the equations below:

$$\dot{x_{err}} = (A - LC)x_{err} \tag{13}$$
$$\dot{x} = (A - BK)x \tag{14}$$

In Equation (12), $x_{err} = \hat{x} - x$, where $\hat{x}$ is the estimated state. The controller doesn't have access to the actual state, so the inputs are calculated using the estimated state instead.

$$u = -K\hat{x} \tag{14}$$

LQR (Linear Quadratic Regulator) is used to design both the observer and the controller. Previously, we used LQR to calculate the K gains in the method shown below. This method is used again to calculate the L gains for the observer, albeit with a lot less effort placed in tuning the observer for accuracy.

To solve for controller gains using the LQR method, we first evaluate the absolute Ricatti equation with A, B, Q_c, and R_c, to get P. Using that value, we can calculate K.

$$K = R_c^{-1}B^T P \tag{15}$$

With these two steps we essentially have a method to implement LQR under one function. This function is created in the Jupyter Notebook as lqr(). It takes in A, B, Q, R as its inputs and gives K.

But since the method is the same for observer design, the same function is used with a few modifications.

- LQR function implementation for controller design:

$$K = lqr(A, B, Q_c, R_c,) \tag{16}$$

- LQR function implementation for observer design:

$$L^T = lqr(A^T, C^T, Q_o, R_o) \tag{17}$$

With the LQR function ready, the only pieces of information required to calculate the controller and observer gains are the cost matrices Q_c, R_c, Q_o, R_o.

E. LQR Controller

To design the controller, some Q_c and R_c values need to be chosen. After tuning these gains to get results that best satisfy the requirements, the final values of the cost matrices are given below:

$$Q_c = diag[0.5, 1, 1, 0.25, 0.4, 0.4] \tag{18}$$
$$R_c = diag[1, 1, 1, 1]$$

Though the ratio between the error and effort weights isn't large, and so the final torques aren't large, these values worked well and helped the system satisfy its requirements. The weights on the roll error are smaller because it was observed that larger weights on the roll led to oscillations, potentially due to overcorrection.

The calculated K matrix is shown below:

[2] This is not proven in this report but was proven in class materials that are acknowledged in the acknowledgement section.

$$K = \begin{bmatrix} -0.5 & 0 & -0.5 & -3.1087 & 0 & -2.607 \\ 0.5 & 0 & -0.5 & 3.1087 & 0 & -2.607 \\ 0 & -0.7071 & -0.5 & 0 & -3.7001 & -2.607 \\ 0 & 0.7071 & -0.5 & 0 & 3.7001 & -2.607 \end{bmatrix} \quad (19)$$

We can calculate the eigenvalues of $F = A - BK$ to check if the system is stable. The eigenvalues calculated in the code had all negative real parts, which meant that the chosen controller gains resulted in a stable system.

F. LQR Observer

Similarly, we design our observer by selecting the Q_o and R_o cost matrices. For observer design, larger Q_o places a greater weight on the sensor model, whereas larger R_o values place a greater weight on the dynamic model. For a system with random initial conditions, putting more weight on the dynamic model caused issue and resulted in large errors between the actual state and the estimated state. Therefore, the only tuning that was done was decreasing the R_o weights.

$$Q_o = diag[1, 1, 1, 1, 1, 1, 1, 1, 1, 1] \quad (20)$$
$$R_o = diag[0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$$

With this setup, we get a value for L. Because L is a large 6×10 matrix, it was not placed in this report.

We can check the stability of the calculated gains by checking the eigenvalues of the matrix $H = A - LC$. The eigenvalues calculated in the code had all negative real parts, indicating that the chosen L results in a stable observer with decreasing error between the state estimate and the state.

We can now run many simulations and see the results of our controller and observer design.

VI. Results

The control and observer are implemented in the pybullet simulation environment. As mentioned before, the initial angular conditions are randomized, whereas the initial angular velocities are zero. There is also space debris that randomly generates and collides with the spacecraft, creating disturbances. The star tracker also has a noise of 0.1 for this simulation.

The system is run 500 times with this setup, with each simulation running for 30 seconds with the controller and observer designed in the previous sections.

The histograms in Figure 3 show the results of the 500 simulations. They show the frequency of the mean squared errors in the roll, pitch, and yaw of the system.

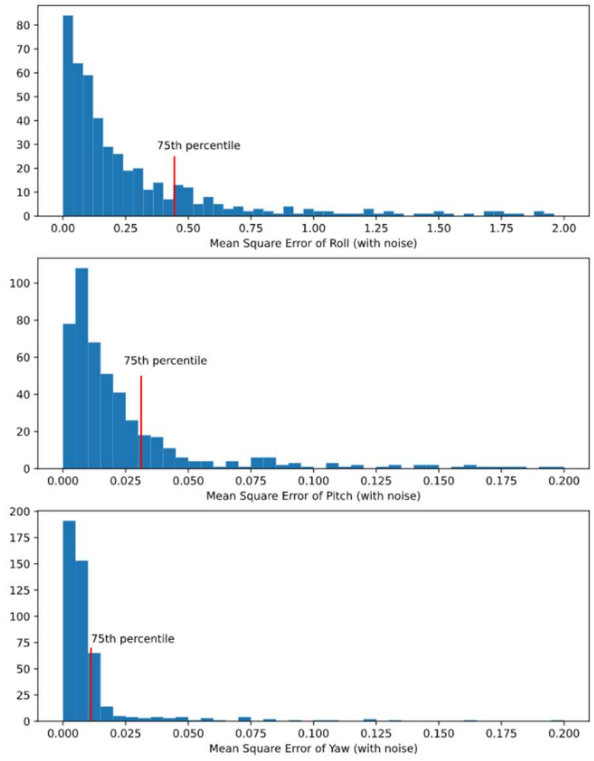

Figure 3: Mean Squared Error Data of the Orientation

Table 1 highlights some of the takeaways from this data, as it provides more insight into the robustness of the system. Special attention is given to the 75th percentile value, which is used as a benchmark in the requirements.

Table 1: Statistics on the mean squared errors

	Roll MSE	Pitch MSE	Yaw MSE
Mean	0.2933	0.0314	0.0671
Median	0.1281	0.0137	0.0051
Maximum	3.7844	0.5120	4.6052
Minimum	0.0025	0.00017	0.000054
Standard Deviation	0.4662	0.0606	0.3993
75th percentiles	0.3445	0.0262	0.0090

We see that the 75th percentile values are in fact much lower than the maximum values set by the requirement. Therefore, we have satisfied the requirement and our system works!

Another interesting point is the number of simulations where the roll mean squared error is less than 0.05, and the pitch and yaw mean squared errors have values less than 0.005. This would be a **very** good result. From the data, we see that this occurs 22 times, i.e. 4.4% of the time. Therefore, 4.4% of the time our controller does **10 times better** than expected in the requirements. That's awesome!

G. Error between estimate and true state

The effectiveness of the controller has already been verified using the data above, which requires both the controller and observer to be functional. However, to further highlight the efficacy of the controller, shown in is a graph of the error between the estimated and true roll, pitch and yaw for a single 50 second random simulation. This particular simulation happens to be one where the observer's estimate is close the the true state. However, this is not

always the case, especially with the random disturbances. For example, below we see that there is a spike in the roll error at a point. This is likely due to some collision. It is best to refer to the previous section for proper data.

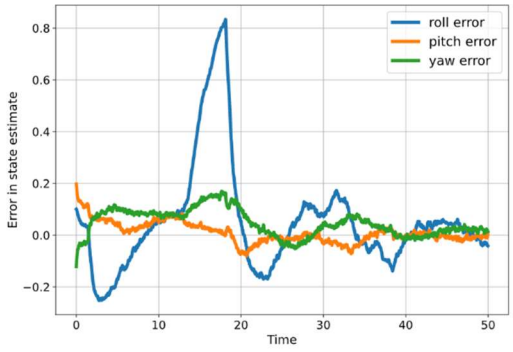

Figure 4: Error in State Estimate over time

VII. Effects of Noise

Real sensors aren't perfect, they are noisy. This means that there are additional internal disturbances in the data of the sensor caused due to the the electrical nature of the device. Although noise is unavoidable and random, our observer design mitigates many of its harsh effects.

To highlight this further, I decided to explore what happens if I increase the sensor noise 3-fold, all the way to a noise setting of 0.3. We can see in the histograms in Figure 5 that the mean square error is larger for sure. However, it is not too large. In fact, the difference would hardly be noticeable in a single simulation run. This speaks to the benefits of using optimal observer control rather than another method such as dead reckoning.

Figure 5 gives the histograms for 500 simulations again, but this time with greater noise.

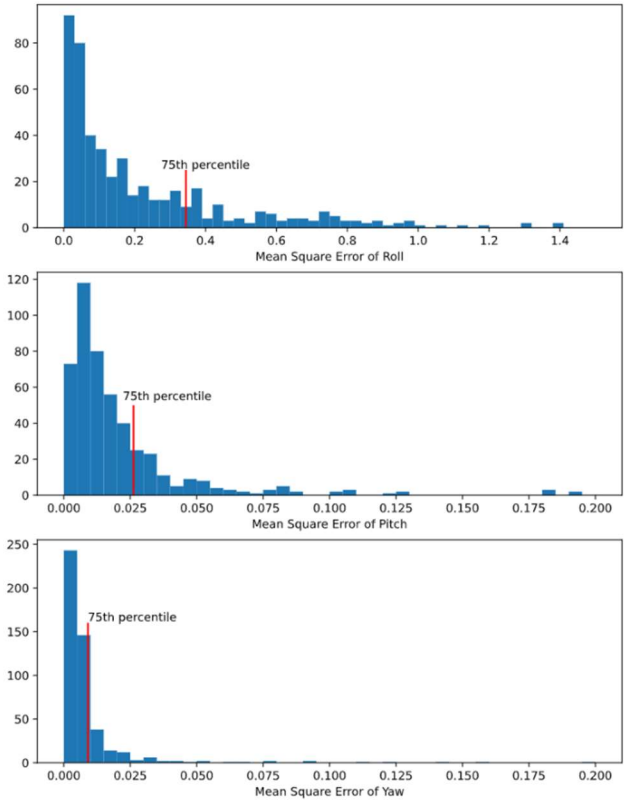

Figure 5: Mean Squared Errors in the Noisy Simulation

Surprisingly, the 75th percentile values *still* satisfy the requirement set. Therefore, even with scope noise at 0.3, our controller and observer work.

I found this topic especially fascinating because in some of the projects that I'm working on now either personally or as part of a student team, we've started working with state estimation. Mainly, with inertial measurement units. Reducing noise can be useful, but designing a good observer means that your system would be mostly unaffected by an increase in noise. Even if the performance is affected, it would only be a small amount compared to how much dead-reckoning's effectiveness falls.

VIII. Conclusion

A satellite simulation was set up with a star tracker that tracks the position of 5 stars. The tracker is noisy, the satellite starts at a random initial orientation and is often hit by space debris. However, for more than a majority of the time, the satellite's control system allows it to maintain an attitude with small errors from the desired orientation.

To verify this system, 500 simulations were run to ensure that the system is actually capable of controlling the satellite and the rocket. In the last few sections, this data was analyzed to ensure that the observer and controller were working. Finally, the system's behavior with more noise was also analyzed, and proved the validity of an observer further.

There is still a lot of room for improvement in the controller and observer design. Though they seem to work fine and satisfy the current requirements, they cannot handle many other conditions (such as randomized angular velocity) reliably.

Acknowledgements

I'd like to gratefully acknowledge the instruction of Professor Timothy Bretl and graduate teaching assistant Jacob Kraft, who provided the guidance and material to successfully complete this project. Their teachings in class, their example code and the homework tutorials served as a backbone to this project. Their guidance on both this project and previous projects was instrumental. I'd also like to acknowledge all my peers whose answers on the discussion forum helped from time to time. I'd also like to acknowledge my own work on the first two design projects, which influenced the way I completed this design project.

References

There are no additional references for this report. Website references are listed in the footnotes.